IN THE SPECIFICATION

Please amend the paragraph at page 1, beginning at line 5 as follows:

This application is related to **and claims priority to U.S. Provisional Patent Application No.** ~~application serial numbers 60/249,627 and~~ 60/264,667 **filed on January 25, 2001, the contents of which** ~~from which this application claims priority under 35 U.S.C. § 119(e), and serial number 09/780,895 based on 35 U.S.C. § 120. All Three of the above applications~~ are incorporated herein by reference.

Please amend the paragraph at page 7, beginning at line 7 as follows:

Each IPv4 packet has a layer 3 field containing a 32-bit destination IP address. In the following embodiments, the most significant 16 bits are grouped together and called a segment 605 and the remaining K bits are called an offset 610. K is variable ranging from 1 to 16 and is chosen in order to minimize redundancy in the table. The data structure will have two levels of tables in order to store the routing information base (RIB): namely, T1_RIB (first level) and T2_RIB (second level) tables. FIG. ~~[[7]]~~ **1** shows the two hierarchical levels for the 16/K data structure. The most significant 16 bits of an IP destination address are used as an index to the T1_RIB table 615. The index to the T1_RIB table 615 ranges from 0.0 (for the first entry 625) to 255.255 (for the last entry 625). Totally the T1_RIB table 615 has 216 entries. Each entry 625 in the T1_RIB table 615 is 4 bytes; thus its total size is 216 * 4 bytes = 256 KB. Each entry 625 in the T1_RIB table 615 stores next hop **630** and prefix length **635** (NHPL) information ~~655~~ if there is not any route whose prefix matches the index of that entry 625, with a prefix length 635 greater than 16. If there are one or more routes associated with that entry 625 having a prefix length 635 greater than 16, that entry 625 instead stores a K value **645** and the base address 640 pointing to a T2_RIB table 620 that has 2K entries. For those entries in the T1_RIB table 615 that store base addresses 640 pointing to a T2_RIB table 620, they will use distinct, unique base addresses 640. The remaining K bits are used in the IP destination address as an offset pointing to a particular entry 625 in the T2_RIB table 620. Each entry 625 in the T2_RIB table 620 is two bytes and stores the NHPL information 655 for a route.

Please amend the paragraph at page 8, beginning at line 7 as follows:

It is necessary to store the prefix length 635 of each route entry 625 for route update. That is, a more specific route will overwrite a less specific route. Suppose the initial route table is empty. If a new IP route 38.0.0.0/8/1 (the first field is the 32-bit IP address in dot format, the second field "8" indicates prefix length 635 while the third field "1" is the next hop 630) arrives. This implies that the T1_RIB table 615 from 38.0 to 38.255 (total 28 = 256 entries) needs to be updated to reflect this newer route. Next, suppose a new IP route 38.170.0.0/16/2 arrives. The entry 625 indexed by 38.170 in the T1_RIB table 615 are overwritten with the new next hop and prefix lengths 2 and 16, respectively. If the order of the two coming routes is reversed, the routing tables would look the same because the less specific route (38.0.0.0/8/1) would not overwrite the more specific route (38.170.0.0/16/2) at the index 38.170 in the T1_RIB. More discussion on how to update the routing table will follow shortly. The format of each entry 625 in the T1_RIB and the T2_RIB tables 620 is shown in FIGs. ~~8A and 8B~~ **2A and 2B**.

Please amend the paragraph at page 8, beginning at line 15 as follows:

For each T1_RIB entry 625, say `T1_Entry[31:0]`, use the bit fields as follows. `T1_Entry[31]` is the most significant bit (a marker bit **650**) and represents whether this entry 625 stores next hop/prefix length information or a K value/pointer to a T2_RIB table 620. If `T1_Entry[31]` is 0, `T1_Entry[30:16]` is not used, `T1_Entry[15:6]` stores next hop information 630 and `T1_Entry[5:0]` stores the prefix length 635 associated with this route. Otherwise, `T1_Entry[30:27]` stores the value **645** of (K-1) (note these 4 bits can represent the value from 0 to 15, thereby indicating the real K value from 1 to 16) and `T1_Entry[26:0]` stores a base pointer **640** to its T2_RIB. These 27 bits are far more than sufficient for indexing into the second level table T2_RIB since the size of the tables created will never require 128 MB (2^{27} bytes) of memory space).

Please amend the paragraph at page 15, beginning at line 7 as follows:

FIGs. 4 and 5 illustrate example ~~This section describes the~~ route lookup and update algorithms**, respectively,** for both the 16/K and 16/Kc data structures. Upon receiving an IPv4

data packet at the ingress line card, the router will take the following steps for each routing table lookup:

Please amend the paragraph at page 16, beginning at line 8 as follows:

Moving on to the 16/K route update algorithm, **as shown in FIG. 5,** upon receiving an IP routing control packet containing the 3-tuple (`Ip_Addr, Prefix_Length, Next_Hop`) (FIG. 5, S750), the routing table needs to be updated. In this application, only how to add a new route is discussed. Deleting a route would take the inverse action and is omitted in this paper for simplicity. To add a new route, one needs to consider two cases: 1) the prefix length 635 associated with the new route is either less than or equal to 16; 2) it is greater than 16.

Please amend the paragraph at page 19, beginning at line 15 as follows:

An example ~~The~~ lookup algorithm for the 16/Kc data structure is described below **in conjunction with the flowchart in FIG. 7**.

Please amend the paragraph at page 21, beginning at line 1 as follows:

The following is a description of ~~the~~ **an example** 16/Kc update algorithm **with reference to FIGs. 8A and 8B**. **As shown in FIG. 8A, when** ~~When~~ a new route needs to be added to the database (FIG. 8A, S903), it will fall under one of two categories, `Prefix_Length <= 16` or `Prefix_Length > 16`.

Please amend the paragraph at page 21, beginning at line 14 as follows:

If `Prefix_Length` > 16 (S906), this corresponds to a single T1_RIB entry 625 indexed by the 16-bit `Ip_Addr[31:16]`. If the T1_RIB entry 625 at this index has its marker bit off (FIG. 8B, S930), then this is an attempt to add a route more specific than the one specified in the T1_RIB entry 625. A new T2_RIB of size 2^{New_K-6} needs to be created, populated with data, and linked into the T1_RIB entry 625 (S942). Specifically, the T2_RIB table 620 is created, all bitmaps 660 are initialized to a single leading 1, and the NHPL array is initialized to contain the NHPL data of the original T1_RIB entry 625. Lastly, the new route is added according to the previously specified rules which results in changing the bitmap 660 and NHPL array. Otherwise

if the marker bit is on (S930), a different procedure**, as illustrated in FIG. 8B,** must be followed to add the entry 625 to the T2_RIB table 620.

Please amend the paragraph at page 21, beginning at line 14 as follows:

As shown in FIG. 8B, if ~~If~~ the entry 625 being added has an `Old_K >= New_K` (S933) then two different cases can be identified -- `Old_K` ≤ 6 and `Old_K` > 6.